

Software–Projekt 1 2013

VAK 03-BA-901.02

Architekturbeschreibung




| | | |
|-------------------|-------------------------------|---------|
| David Brinkmann | david.brinkmann@uni-bremen.de | 2696099 |
| Patrick Damrow | damsen@tzi.de | 2056170 |
| Daniel Pupat | s_aydi4h@uni-bremen.de | 2703053 |
| Michael Sauerwein | s_rfkrpo@uni-bremen.de | 2699739 |
| Leopold Siakeu | henrileopold@yahoo.fr | 2193984 |

Inhaltsverzeichnis

| | | |
|----------|--|-----------|
| 1 | Einführung | 3 |
| 1.1 | Definitionen, Akronyme und Abkürzungen | 3 |
| 1.2 | Referenzen | 3 |
| 2 | Globale Analyse | 4 |
| 2.1 | Einflussfaktoren | 4 |
| 2.2 | Probleme und Strategien | 11 |
| 3 | Konzeptionelle Sicht | 18 |
| 4 | Modulsicht | 20 |
| 5 | Datensicht | 23 |
| 6 | Zusammenhänge zwischen Anwendungsfällen und Architektur | 24 |

Version und Änderungsgeschichte

| Version | Datum | Änderungen | |
|---------|------------|--|---|
| 1.0 | 01.06.2013 | Dokumentvorlage als initiale Fassung kopiert |  |
| 1.1 | 05.06.2013 | Einflussfaktoren | |
| 1.2 | 07.06.2013 | Probleme und Strategien | |
| 1.3 | 12.06.2013 | Konzeptionelle Sicht, Modulsicht | |
| 1.4 | 14.06.2013 | Datensicht, Zusammenhänge Anwendungsfällen und Architektur | |
| 1.5 | 24.06.2013 | Sequenzdiagramm und Beschreibung (Punkt 6 Zusammenhänge) | |
| 1.6 | 26.06.2013 | Referenzen, Definitionen | |
| 1.7 | 26.06.2013 | Formatierung und Fertigstellung | |

1 Einführung

1.1 Definitionen, Akronyme und Abkürzungen

(David Brinkmann)

- **GUI** (graphical user interface - grafische Benutzeroberfläche): Die Anzeige, mit der der Benutzer interagieren kann.
- **user-ID**: Eine eindeutige Nummer, welche einem Nutzer während einer Sitzung zugewiesen wird und anhand welcher er identifiziert wird.

1.2 Referenzen

(David Brinkmann)

- IEEE Std 830-1998 Recommended Practice for Software Requirements Specifications
- IEEE Standard 1058-1998 for Software Project Management Plans
- Architekturspezifikation-Beispiel (PDF, StudIP - Software-Projekt 1 Praktikum)
- Hinweise Architekturspezifikation (PDF, StudIP - Software-Projekt 1 Praktikum)
- Vorlesungsfolien SWP-1, Rainer Koschke

2 Globale Analyse

2.1 Einflussfaktoren




(David Brinkmann)

Tabelle 1: Technische Faktoren

| Nummer | Faktoren |
|--------|--|
| 1 | Kompatibilität auf Endgeräten (Android 2.3.x+) |
| 2 | Bilingualität |
| 3 | Backup und Wiederherstellung |
| 4 | Abspeichern und auslesen von Daten |
| 5 | Administratoraccount |
| 6 | Kompatibilität des Servers für Windows und Linux |
| 7 | Beschränkungsfreiheit der Fremdbibliotheken |
| 8 | Testbarkeit |
| 9 | Zuverlässigkeit |
| 10 | Performanz |
| 11 | Verwendung einer SQL-Datenbank |

| | |
|---|-------------------------------|
| 1 | Kompatibilität auf Endgeräten |
| Auswirkungen <ul style="list-style-type: none">• Die Applikation muss auf den geforderten Endgeräten (Smartphone sowie Tablet) lauffähig sein• Möglichkeit der Anpassung an gerätespezifische Funktionen (Bildschirmgröße, Auflösung) | |
| Flexibilität und Veränderlichkeit <ul style="list-style-type: none">• Möglichkeit der Erweiterung des uns gegebenen Framework• Aufgrund der Vorgaben keine Änderungen vorgesehen | |

| | |
|--|---------------|
| 2 | Bilingualität |
| Auswirkungen <ul style="list-style-type: none">• Bei der Implementierung muss auf die Notwendigkeit zweier Sprachen geachtet werden | |
| Flexibilität und Veränderlichkeit <ul style="list-style-type: none">• Keine Flexibilität oder Veränderlichkeit, da fest vorgegeben | |

| | |
|--|------------------------------|
| 3 | Backup und Wiederherstellung |
| Auswirkungen  <ul style="list-style-type: none">• Das Programm muss diese Möglichkeit der Datensicherung bieten• Die Art der Sicherung muss an die gegebenen Umstände angepasst werden (Welche Medien sind vorhanden) | |
| Flexibilität und Veränderlichkeit <ul style="list-style-type: none">• Möglichkeit eine automatischen Backups oder der manuellen Durchführung durch den Admin• Beide Möglichkeiten sollten vorhanden sein | |

| | |
|---|------------------------------------|
| 4 | Abspeichern und auslesen von Daten |
| Auswirkungen <ul style="list-style-type: none">• Server übernimmt die Speicherung der Daten• Client und Mobile Endgeräte erhalten ihre Daten per Internet/Intranet vom Server• Auf den Endgeräten werden keine Daten dauerhaft gespeichert | |
| Flexibilität und Veränderlichkeit <ul style="list-style-type: none">• Keine | |


| | |
|--|----------------------|
| 5 | Administratoraccount |
| Auswirkungen | |
| <ul style="list-style-type: none">• Implementierung des Administratoraccounts als Benutzerkonto mit besonderen Rechten | |
| Flexibilität und Veränderlichkeit | |
| <ul style="list-style-type: none">• keine Flexibilität oder Veränderlichkeit, da fest vorgegeben | |

| | |
|--|--|
| 6 | Kompatibilität des Servers für Linux und Windows |
| Auswirkungen | |
| <ul style="list-style-type: none">• Die Software muss auf beiden Systemen voll einsatzfähig sein und den Anforderungen entsprechen | |
| Flexibilität und Veränderlichkeit | |
| <ul style="list-style-type: none">• keine Flexibilität oder Veränderlichkeit, da fest vorgegeben | |

| | |
|--|---|
| 7 | Beschränkungsfreiheit der Fremdbibliotheken |
| Auswirkungen | |
| <ul style="list-style-type: none">• Wir müssen uns rechtzeitig auf ein Version einigen• Die Ausgewählte Bibliothek muss den Anforderungen entsprechen | |
| Flexibilität und Veränderlichkeit | |
| <ul style="list-style-type: none">• Feste Vorgabe | |

| | |
|---|-------------|
| 8 | Testbarkeit |
| Auswirkungen | |
| <ul style="list-style-type: none">• Vor Beginn der Implementierung müssen aussagekräftige Tests ausgearbeitet werden• Die Ausführung der Tests muss vor Abgabe des Produkts erfolgen | |
| Flexibilität und Veränderlichkeit | |
| <ul style="list-style-type: none">• Die Art der Tests ist frei von uns wählbar• Tests als solche fest vorgegeben | |

| | |
|--|-----------------|
| 9 | Zuverlässigkeit |
| Auswirkungen | |
| <ul style="list-style-type: none">• Die Programmierarbeit erfolgt nach geltenden Regeln und Vorgaben• Die Zuverlässigkeit muss durch mehrere Tests sichergestellt werden• Ein mögliches Zeitfenster für Verbesserungen muss einkalkuliert werden | |
| Flexibilität und Veränderlichkeit | |
| <ul style="list-style-type: none">• Keine , die Zuverlässigkeit des Produkts ist selbstverständlich | |

| | |
|---|------------|
| 10 | Performanz |
| Auswirkungen  <ul style="list-style-type: none"> • Die Art der Implementierung muss sich nach der besten Performanz des Produktes, bei hoher Zuverlässigkeit, richten • Aussagekräftige Tests müssen erarbeitet werden, um eine ausreichende Performanz zu überprüfen | |
| Flexibilität und Veränderlichkeit <ul style="list-style-type: none"> • Kaum Flexibilität, die Software muss dem Nutzer ein unbeschwertes Arbeiten ermöglichen | |

| | |
|---|--------------------------------|
| 11 | Verwendung einer SQL-Datenbank |
| Auswirkungen <ul style="list-style-type: none"> • Die Software muss für die Verwendung von SQL vorbereitet sein | |
| Flexibilität und Veränderlichkeit <ul style="list-style-type: none"> • Feste Vorgabe | |

Tabelle 2: Organisatorische Faktoren

| Nummer | Faktoren |
|--------|--|
| 12 | Rechtzeitige Abgabe (time-to-market) |
| 13 | Keine Budgetüberschreitung |
| 14 | Kenntnisse von Java und Android |
| 15 | Fachwissen und Anzahl der Teammitglieder |
| 16 | Softwaretests |

| | |
|---|---------------------|
| 12 | Rechtzeitige Abgabe |
| Auswirkungen <ul style="list-style-type: none">• Die Software muss bis zum Abgabetermin fertiggestellt und lauffähig sein. | |
| Flexibilität und Veränderlichkeit <ul style="list-style-type: none">• Die Deadline kann nicht verändert werden und ist somit eine feste Größe. | |

| | |
|---|----------------------------|
| 13 | Keine Budgetüberschreitung |
| Auswirkungen <ul style="list-style-type: none">• Bei Budget handelt es sich lediglich um ein zeitliches Budget. Dieser Faktor wirkt sich daher nur auf unsere Zeiteinteilung aus. | |
| Flexibilität und Veränderlichkeit <ul style="list-style-type: none">• Wenig flexibel• Da wir bereits fast das Maximum unserer Zeit aufwenden, ist die Flexibilität eingeschränkt.• Zeitliches Budget kann nur vergrößert werden, wenn andere Vorlesungen/Tutorien/Abgaben vernachlässigt werden. | |

| | |
|---|---------------------------------|
| 14 | Kenntnisse von Java und Android |
| Auswirkungen <ul style="list-style-type: none">• Es ist ein gewisser Kenntnisstand notwendig, um die Software implementieren zu können. | |
| Flexibilität und Veränderlichkeit <ul style="list-style-type: none">• In der kurzen Zeit ist es nicht möglich, den Kenntnisstand großartig zu erweitern. Ebenso wenig ist es möglich, neue Mitglieder in die Gruppe aufzunehmen und damit mangelnde Kenntnisse zu kompensieren. Daher ist dieser Faktor sehr unflexibel. | |

| | |
|---|--|
| 15 | Fachwissen und Anzahl der Teammitglieder |
| Auswirkungen <ul style="list-style-type: none">• Abgesehen von Punkt 14 werden auch weitere Kenntnisse benötigt. Außerdem wird mit einer festen Anzahl aus 6 Teammitgliedern gerechnet. | |
| Flexibilität und Veränderlichkeit <ul style="list-style-type: none">• Teammitglieder können in der verbleibenden kurzen Zeit nicht mehr aufgenommen werden. Ebenso ist es nur schwer möglich, mit dem geringen zeitlichen Budget neue Fachkenntnisse zu erwerben.• Unbekannt ist, wie gut und schnell das Wissen in den noch kommenden Blockkursen (Datenbanken) erworben wird. Dies könnte den Wissenstand des kompletten Teams beeinflussen.• Ein Mitglied wurde bereits aus dem Team ausgeschlossen. Weitere Verluste in dieser Hinsicht könnten ebenfalls das Voranschreiten des Projekts behindern. | |

| | |
|--|---------------|
| 16 | Softwaretests |
| Auswirkungen <ul style="list-style-type: none">• Es müssen unterschiedliche Tests durchgeführt werden• Bei Abgabe muss das Projekt alle notwendigen Tests vorweisen können | |
| Flexibilität und Veränderlichkeit <ul style="list-style-type: none">• Da Tests obligatorisch sind, ist dieser Faktor nicht flexibel. | |

2.2 Probleme und Strategien


(Daniel Papat)

Tabelle 3: Probleme


| Nummer | Faktoren |
|--------|--|
| 1 | Mangelnde Programmierkenntnisse |
| 2 | Unterschiedlicher Programmierstil |
| 3 | Unzureichende Softwaretest |
| 4 | Einhaltung der geforderten Software-Performanz |
| 5 | Probleme bei der Textübersetzung |
| 6 | Einhalten der zeitlichen Vorgabe |
| 7 | Personalausfall |



| | |
|---|---------------------------------|
| 1 | Mangelnde Programmierkenntnisse |
| <ul style="list-style-type: none">• Mangelnde Programmierkenntnisse sind ein schwerwiegendes Problem. Ohne die notwendigen Kenntnisse kann das Projekt nicht realisiert werden. Dieses gilt für Java, sowie auch für Android und dem SQL-Datenbanksystem. | |
| Einflussfaktoren <ul style="list-style-type: none">• T1: Kompatibilität auf Endgeräten (Android 2.3.x+)• T4: Abspeichern und auslesen von Daten• T5: Administratoraccount• T6: Kompatibilität des Servers für Windows und Linux• T8: Testbarkeit• T9: Zuverlässigkeit• T10: Performanz• T11: Verwendung einer SQL-Datenbank• O14: Kenntnisse von Java und Android• O16: Softwaretests | |
| Lösung <ul style="list-style-type: none">• Strategie 1: Bibliotheken nutzen Es werden wenn möglich bereits vorhandene Java-Bibliotheken verwendet, wodurch Code weniger wird und es zu weniger Fehlern kommt.• Strategie 2: Implementierung aufteilen Die Implementierung wird so unter den Mitgliedern verteilt, dass wenn möglich jeder einen Teil Implementiert in dem er bereits ein Vorwissen aufzuweisen hat. | |


| | |
|--|-----------------------------------|
| 2 | Unterschiedlicher Programmierstil |
| <ul style="list-style-type: none">• Da in diesem Projekt in einer größeren Gruppe programmiert wird und jeder einen eigenen Programmierstil hat, kann es dort zu Problemen kommen. Zum einen kann es zu Problemen beim Verständnis unter den Mitgliedern kommen, zum anderen würden Dritte Probleme beim flüssigen lesen bekommen, da sie sich immer auf einen anderen Stil einstellen müssen. | |
| Einflussfaktoren <ul style="list-style-type: none">• O14: Kenntnisse von Java und Android• O15: Fachwissen und Anzahl der Teammitglieder• O16: Softwaretests | |
| Lösung <div></div> <ul style="list-style-type: none">• Strategie 1: Auf einen Stil einigen Unter den Mitgliedern wird vorher abgesprochen wie programmiert wird, dass sollte sich grob an die Norm richten, da so ein guter Mittelweg gefunden werden kann | |


| | |
|---|-----------------------------|
| 3 | Unzureichende Softwaretests |
| <ul style="list-style-type: none">• Bei unzureichenden Softwaretests können in der Implementierungsphase erst spät Fehler bemerkt werden, wodurch es zu vielen Änderungen kommen kann. Die Tests dienen auch dazu die Software später zu testen, ob alles funktioniert. | |
| Einflussfaktoren <ul style="list-style-type: none">• T4: Abspeichern und auslesen von Daten• T8: Testbarkeit• T9: Zuverlässigkeit• T10: Performanz• O14: Kenntnisse von Java und Android• O16: Softwaretests | |
| Lösung <ul style="list-style-type: none">• Strategie 1: Tests während der Implementierung überprüfen Falls Änderungen in der Implementierungsphase auftreten die Tests aktualisieren, damit der spätere Code auch getestet wird.• Strategie 2: Lieber mehr Test schreiben als zu wenig Wenn man zu dem Punkt kommt, wo man unsicher wird, ob man noch weitere Tests schreiben soll, lieber zu viele schreiben als zu wenig. | |

| | |
|---|--|
| 4 | Einhaltung der geforderten Software-Performanz |
| <ul style="list-style-type: none">• Die Software muss eine angemessene Performanz auf allen Endgeräten wie Smartphones aufweisen, so wie es der Kunde gefordert hat. | |
| Einflussfaktoren <ul style="list-style-type: none">• T1: Kompatibilität auf Endgeräten (Android 2.3.x+)• T4: Abspeichern und auslesen von Daten• T6: Kompatibilität des Servers für Windows und Linux• T9: Zuverlässigkeit• T10: Performanz• T11: Verwendung einer SQL-Datenbank | |
| Lösung <div></div> <ul style="list-style-type: none">• Strategie 1: Tests Die Software in Hinsicht auf die Performanz testen, auf den möglichen Endgeräten und Versionen(Android). Hierbei müssen in erster Linie Rechenschwächere Endgeräte getestet werden, da wenn diese bereits eine gute Performanz haben rechenstarke Endgeräte meistens dann auch eine gute aufweisen. | |

| | |
|--|----------------------------------|
| 5 | Probleme bei der Textübersetzung |
| <ul style="list-style-type: none">• Da die Software in Englisch und in Deutsch sein soll, kann es da zu Problemen kommen, wenn z.B. Gruppenmitglieder keine guten Englischkenntnisse haben. So könnte es bei späteren verwenden des Programms zu Missverständnissen und Verwirrungen der Anwender kommen. | |
| Einflussfaktoren <ul style="list-style-type: none">• T2: Bilingualität• O15: Fachwissen und Anzahl der Teammitglieder | |
| Lösung <ul style="list-style-type: none">• Strategie 1: Übersetzung von einem Gruppenmitglied mit guten Englischkenntnissen machen lassen Im Vorfeld schon mit den Gruppenmitgliedern absprechen, wer gute Englischkenntnisse hat und dann dem Mitglied mit den besten Englischkenntnissen die Übersetzung machen lassen.• Strategie 2: Dritte testen lassen Dritte den englischen Text zeigen und so überprüfen, ob diese es richtig verstehen und dann ggf. ändern | |



| | |
|---|----------------------------------|
| 6 | Einhalten der zeitlichen Vorgabe |
| <ul style="list-style-type: none">• Da es durch SWP eine zeitliche Vorgabe gibt ist diese Einzuhalten und somit begrenzt. Ebenso sind die Ressourcen begrenzt. | |
| Einflussfaktoren <ul style="list-style-type: none">• O12: Rechtzeitige Abgabe (time-to-market)• O15: Fachwissen und Anzahl der Teammitglieder | |
| Lösung <div></div> <ul style="list-style-type: none">• Strategie 1: System schrittweise aufbauen Das System schrittweise aufbauen und die wichtigsten Funktionen zuerst implementieren, damit das System funktioniert und erst mal die wichtigsten Faktoren implementiert sind. Falls dann noch Zeit ist können zusätzliche Funktionen eingefügt werden.• Strategie 2: Verwenden von Java Bibliotheken Bereits vorhandene Java Bibliotheken verwenden und nicht neu schreiben, da so Zeit eingespart werden kann | |
| Verwandte Strategien <ul style="list-style-type: none">• Strategie Bibliotheken nutzen 2.2• Beim nutzen von Bibliotheken spart man Zeit und man macht weniger Fehler beim Programmieren und kann so mangelnde Programmierkenntnisse verringern. | |

| | |
|--|-----------------|
| 7 | Personalausfall |
| <ul style="list-style-type: none">• Personalausfall kann durch Krankheiten, Unfälle oder Studien- bzw. Kursabbruch kommen. Dadurch kann es dann zu zeitlichen Engpässen kommen, da jede Person dann ein Teil der Arbeit der ausfallenden Person übernehmen muss. | |
| Einflussfaktoren <ul style="list-style-type: none">• O12: Rechtzeitige Abgabe (time-to-market)• O15: Fachwissen und Anzahl der Teammitglieder | |
| Lösung  <ul style="list-style-type: none">• Strategie 1: Bei Ausfall früh die anderen Mitglieder benachrichtigen Die Gruppenmitglieder sollten so früh wie möglich Bescheid geben, falls sie irgendwie ausfallen, damit die anderen Mitglieder sich auf die anstehende Mehrarbeit einstellen können. So ist es noch möglich den Teil des anderen zu übernehmen, um die Implementierung rechtzeitig fertig zu bekommen. | |

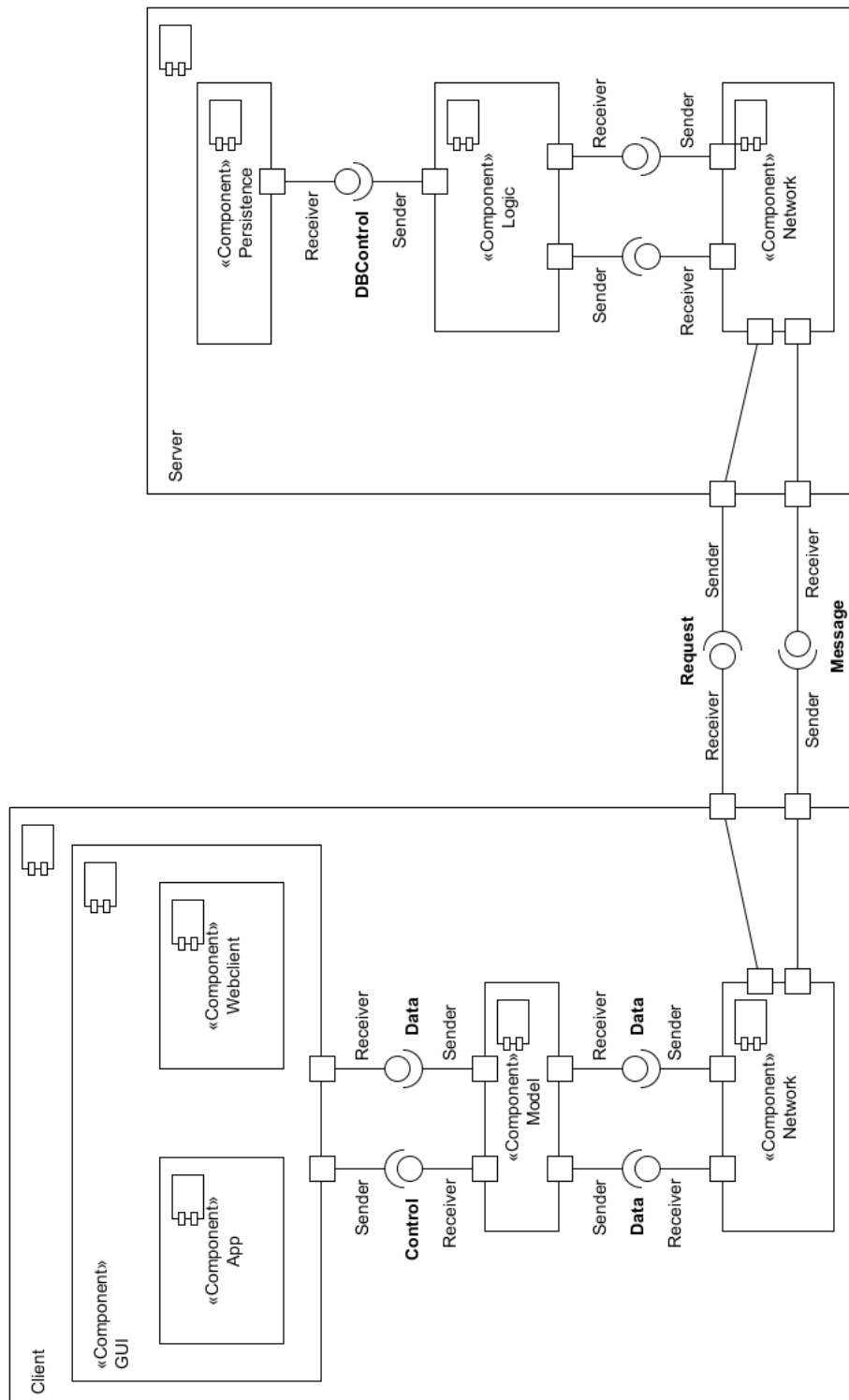
3 Konzeptionelle Sicht

(alle Gruppenmitglieder)

Die Konzeptionelle Sicht wird mittels der folgenden Diagramme dargestellt (1). Insgesamt unterscheiden wir zwischen der Sicht auf den Server und den Client, die mittels der Komponente **Network** miteinander kommunizieren. Die clientseitige GUI-Komponente ist wiederum in zwei Unterkomponenten aufgeteilt. Einerseits gibt es eine GUI, die speziell für die App, und andererseits eine, die für den Webclient, entwickelt wird. `textttModel` ist die verarbeitende, clientseitige Komponente. Dort werden z.B. Benutzereingaben verarbeitet, Cookies gespeichert, etc. Außerdem stellt sie die Verbindung zur Komponente **Network** her.

Der Server kommuniziert mit den verschiedenen Clients über die Komponente **Network**. Die **Logic**-Komponente verarbeitet dabei ankommende Befehle und leitet diese ggf. an den **DBController** weiter. Dieser verwaltet die Zugriffe auf die Datenbank.

Abbildung 1: Konzeptionelle Sicht

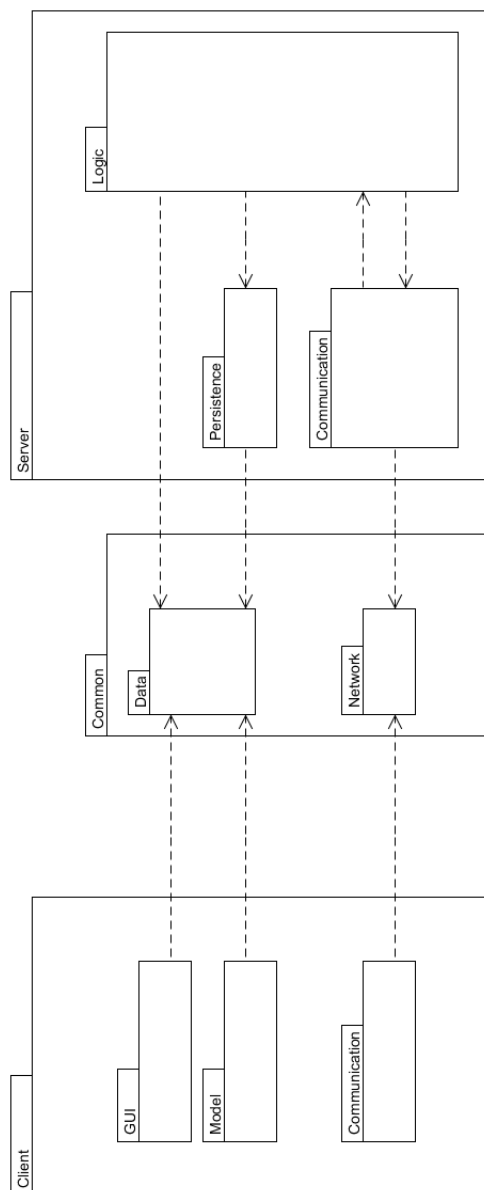


4 Modulsicht

(alle Gruppenmitglieder)

In der Modulsicht werden Klasse und ihre Hauptmethoden dargestellt. Auch hier wird eine Einteilung in **Client** und **Server** verwendet. Abbildung 2 zeigt eine grobe Übersicht.

Abbildung 2: Modulsicht

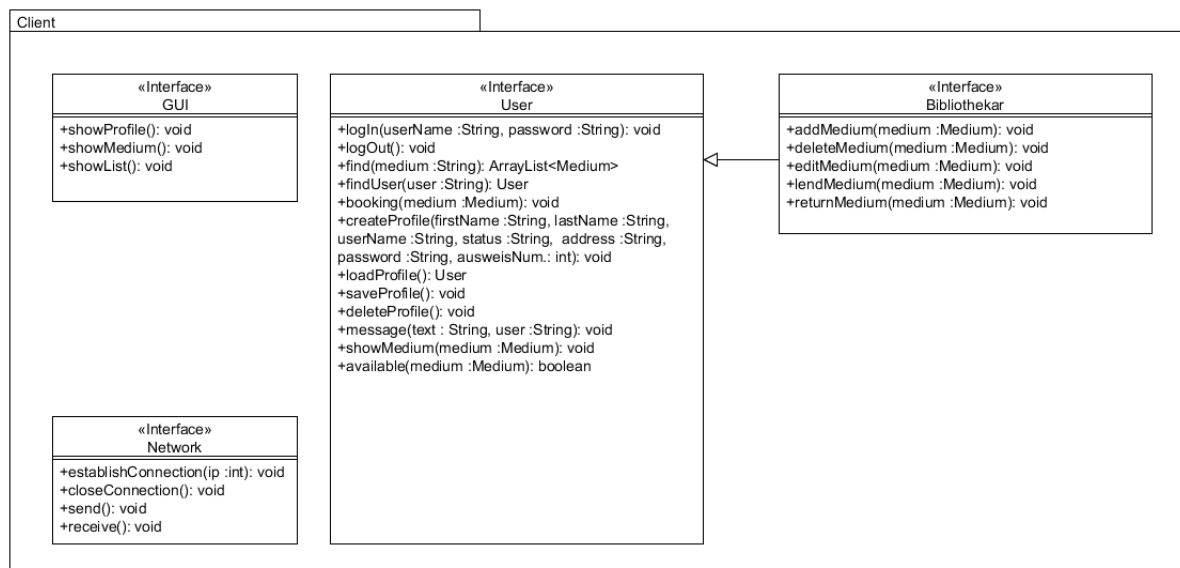


Im Client werden 4 Interfaces implementiert.

- **GUI** Enthält nur Methoden zur Anzeige des Profils, der Leste aller Medien und der Medien selbst.
- **User** enthält alle Methoden, die den Nutzer selbst betreffen. Darunter LogIn und -Out, Medien suchen und anzeigen lassen.
- **Bibliothekar** erbt von User, enthält jedoch weitere Methoden zum Hinzufügen, Editieren und Löschen von Medien.
- **Network** enthält Methoden zum Aufbau einer Verbindung zum Server, Senden und Empfangen von Daten.

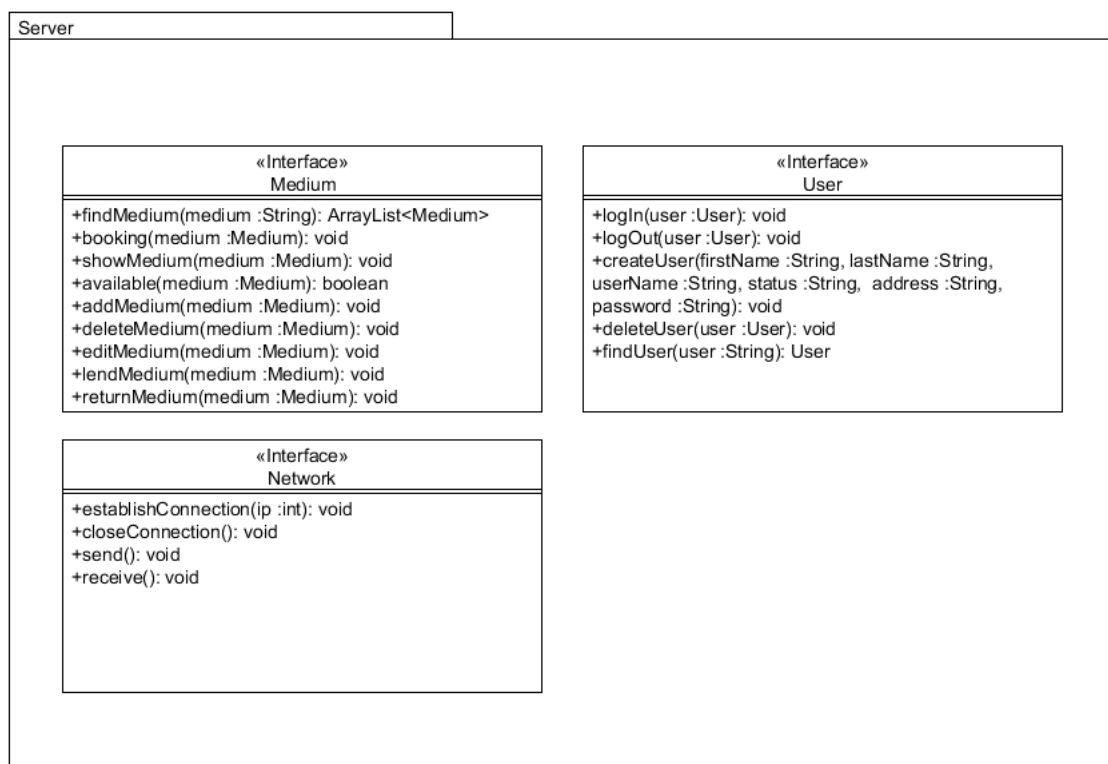


Abbildung 3: Modulsicht Client



- **User** enthält alle Methoden, die den Nutzer selbst betreffen. Darunter LogIn und -Out, Medien suchen und anzeigen lassen.
- **Network** enthält Methoden zum Aufbau einer Verbindung zum Server, Senden und Empfangen von Daten.
- **Medium** enthält alle Methoden welche die Erstellung und Verwaltung von Medien betreffen.

Abbildung 4: Modulsicht Server

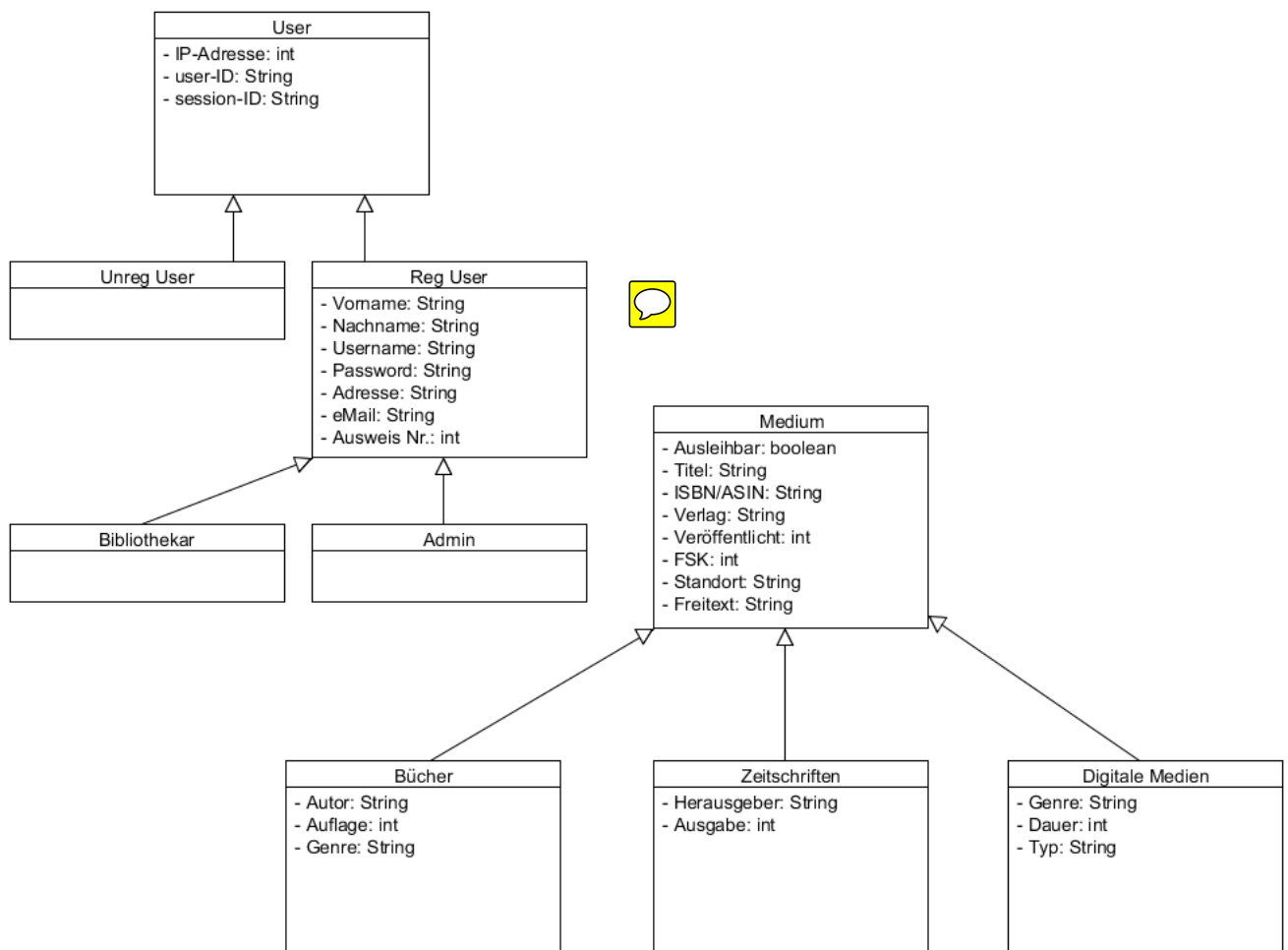


5 Datensicht

(alle Gruppenmitglieder)

Jeder Benutzer des Systems wird anhand seiner User-Id und seiner Session-ID im System identifiziert. Registrierte Benutzer verfügen über erweiterte Rechte, nachdem ihre eingegebenen Nutzerdaten akzeptiert wurden. Je nach ihren Zugriffsrechten, sind sie in der Lage, Objekte der Klasse Medium zu verändern.

Abbildung 5: Datensicht



6 Zusammenhänge zwischen Anwendungsfällen und Architektur

(David Brinkmann)

Anwendungsfall Buch ausleihen:

Abbildung 6 zeigt das Sequenzdiagramm für das Ausleihen eines Buches. Das vom Nutzer gesuchte Buch wird mittels der Methode *find()* an das System geschickt. Dieses liefert anschließend eine Liste aller Bücher zurück, auf die die Suchanfrage passt. Daraufhin sucht sich der Nutzer eines der Bücher aus der Liste aus (*showMedium()*) und erhält detaillierte Informationen zu seinem gewählten Buch. Um das Buch ausleihen zu können, muss sich der Benutzer einloggen. Hat dieser die Einlog-Prozedur hinter sich gebracht, wird mittels *booking()* das Medium für ihn gebucht. Via *logOut()* kann sich nun der Nutzer vom System abmelden.

Abbildung 6: Sequenzdiagramm Buch ausleihen

